Practical Machine Learning Course Project

Project Assignment

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Goal

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. Any of the other variables may be used to predict with. A report should be created describing how the model is built, how cross validation is used, what is the expected out of sample error is, and the choices are made which are done. This prediction model will also be used your predict 20 different test cases.

Reading and loading data

```
set.seed(9999)

training <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
testing <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
```

Divide training set into two partitions

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.2
```

```
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]
myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

Cleaning data

Remove NearZeroVariance variables

```
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
myTraining <- myTraining[,nzv$nzv==FALSE]

nzv<- nearZeroVar(myTesting,saveMetrics=TRUE)
myTesting <- myTesting[,nzv$nzv==FALSE]
```

Remove the first column of the myTraining data set

```
myTraining <- myTraining[c(-1)]
```

Clean variables with more than 60% NA

```
trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
    if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .7) {
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) ==
1)  {
                trainingV3 <- trainingV3[ , -j]
            }
        }
    }
}

# Set back to the original variable name
myTraining <- trainingV3
rm(trainingV3)
```

Convert testing data sets

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58])  # remove the classe column
myTesting <- myTesting[clean1]         # allow only variables in myTesting
that are also in myTraining
testing <- testing[clean2]             # allow only variables in testing that
are also in myTraining

dim(myTesting)

## [1] 7846   58

dim(testing)

## [1] 20 57
```

Coerce data into same type

```
for (i in 1:length(testing) ) {
    for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}

# To get the same class between testing and myTraining
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

Prediction with multiple models for cross validation

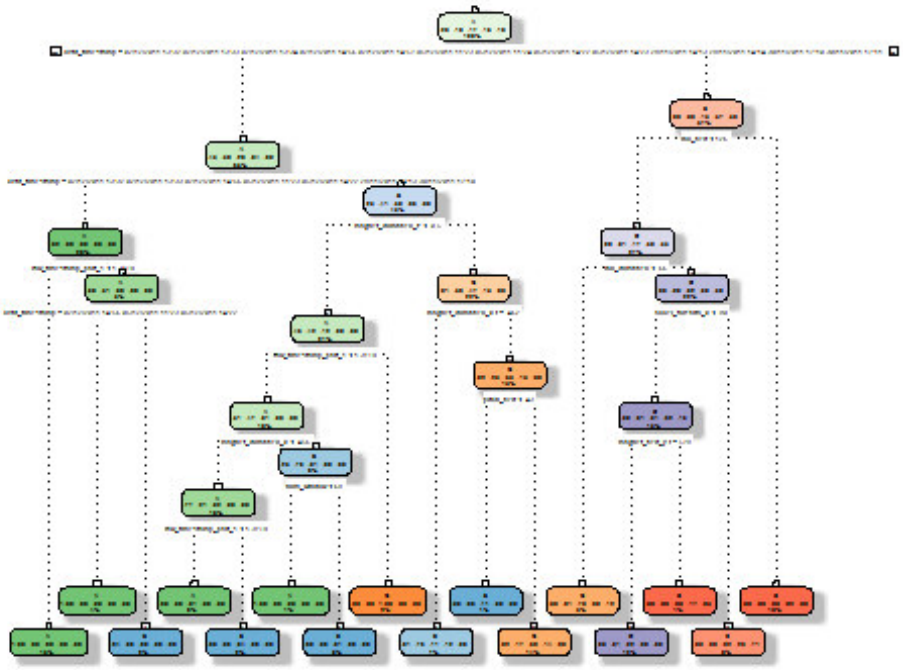
1) Prediction with Decision Trees

```
library(rpart)
library(rattle)

## Warning: package 'rattle' was built under R version 3.2.3

## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

set.seed(9999)
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1, sub="Decision Tree")
```



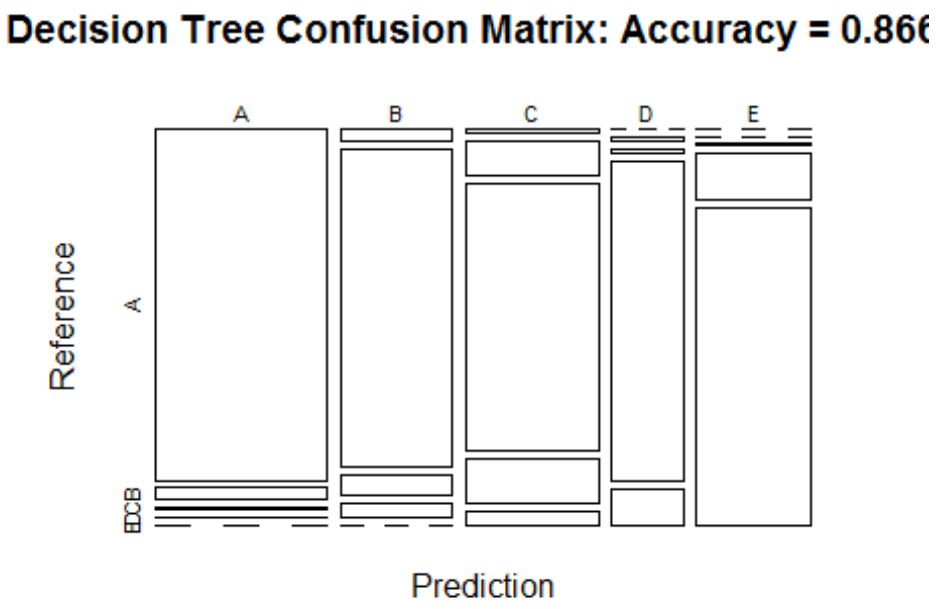
Decision Tree

```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
cmtree <- confusionMatrix(predictionsA1, myTesting$classe)
cmtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2157   71    8    2    0
##          B   55 1265   82   55    0
##          C   20  171 1261  214   61
##          D    0   11   12  824   91
##          E    0    0    5  191 1290
##
## Overall Statistics
##
##                Accuracy : 0.8663
##                  95% CI : (0.8586, 0.8738)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8308
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9664   0.8333   0.9218   0.6407   0.8946
## Specificity            0.9856   0.9697   0.9281   0.9826   0.9694
## Pos Pred Value         0.9638   0.8682   0.7302   0.8785   0.8681
## Neg Pred Value         0.9866   0.9604   0.9825   0.9331   0.9761
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2749   0.1612   0.1607   0.1050   0.1644
## Detection Prevalence   0.2852   0.1857   0.2201   0.1196   0.1894
## Balanced Accuracy      0.9760   0.9015   0.9249   0.8117   0.9320
```

```
plot(cmtree$table, col = cmtree$byClass, main = paste("Decision Tree
Confusion Matrix: Accuracy =", round(cmtree$overall['Accuracy'], 4)))
```



2) Prediction with Random Forests

```
library(randomForest)
```

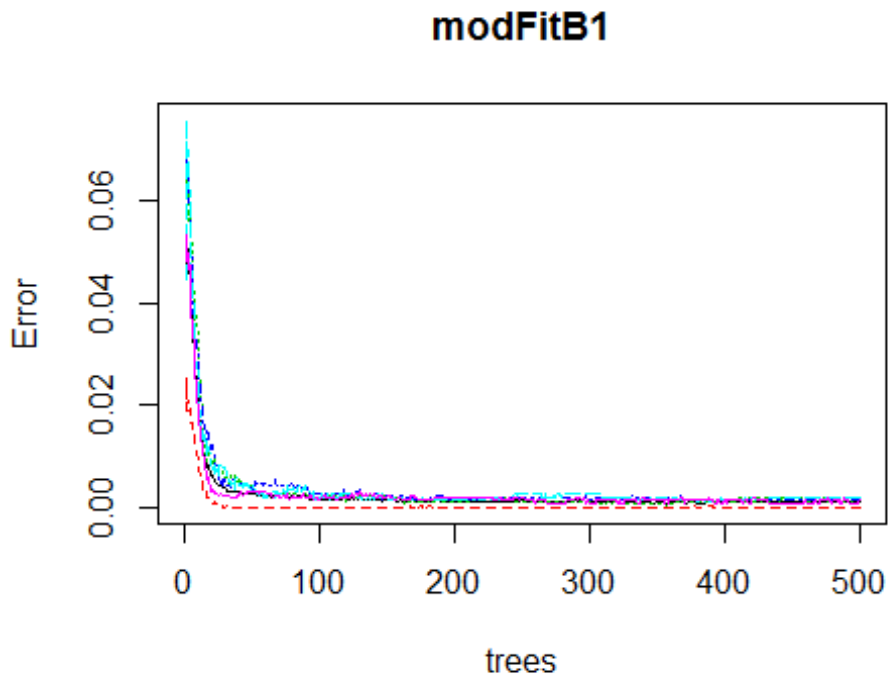
```
## Warning: package 'randomForest' was built under R version 3.2.3
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

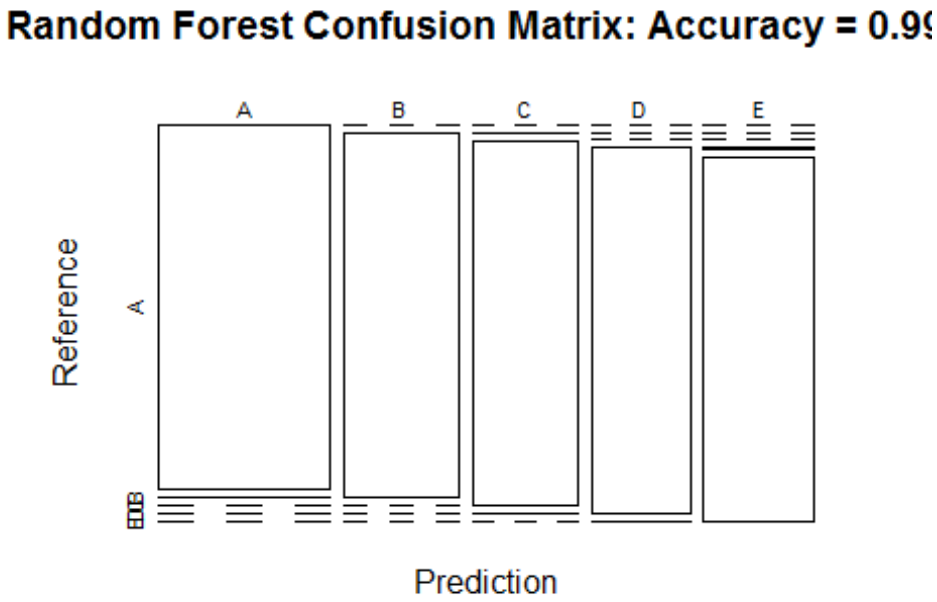
```
set.seed(9999)
modFitB1 <- randomForest(classe ~ ., data=myTraining)
predictionB1 <- predict(modFitB1, myTesting, type = "class")
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1515    0    0    0
##          C    0    1 1368    1    0
##          D    0    0    0 1282    1
##          E    0    0    0    3 1441
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.998, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9980   1.0000   0.9969   0.9993
## Specificity            0.9996   1.0000   0.9997   0.9998   0.9995
## Pos Pred Value         0.9991   1.0000   0.9985   0.9992   0.9979
## Neg Pred Value         1.0000   0.9995   1.0000   0.9994   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1931   0.1744   0.1634   0.1837
## Detection Prevalence   0.2847   0.1931   0.1746   0.1635   0.1840
## Balanced Accuracy      0.9998   0.9990   0.9998   0.9984   0.9994
```

```
plot(modFitB1)
```




```
plot(cmrf$table, col = cmtree$byClass, main = paste("Random Forest Confusion
Matrix: Accuracy =", round(cmrf$overall['Accuracy'], 4)))
```



3) Prediction with Generalized Boosted Regression

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.2.3
```

```
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
library(caret)

set.seed(9999)
fitControl <- trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 1)

gbmFit1 <- train(classe ~ ., data=myTraining, method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE)
```

```
## Loading required package: plyr
```

```
## Warning: package 'plyr' was built under R version 3.2.2
```

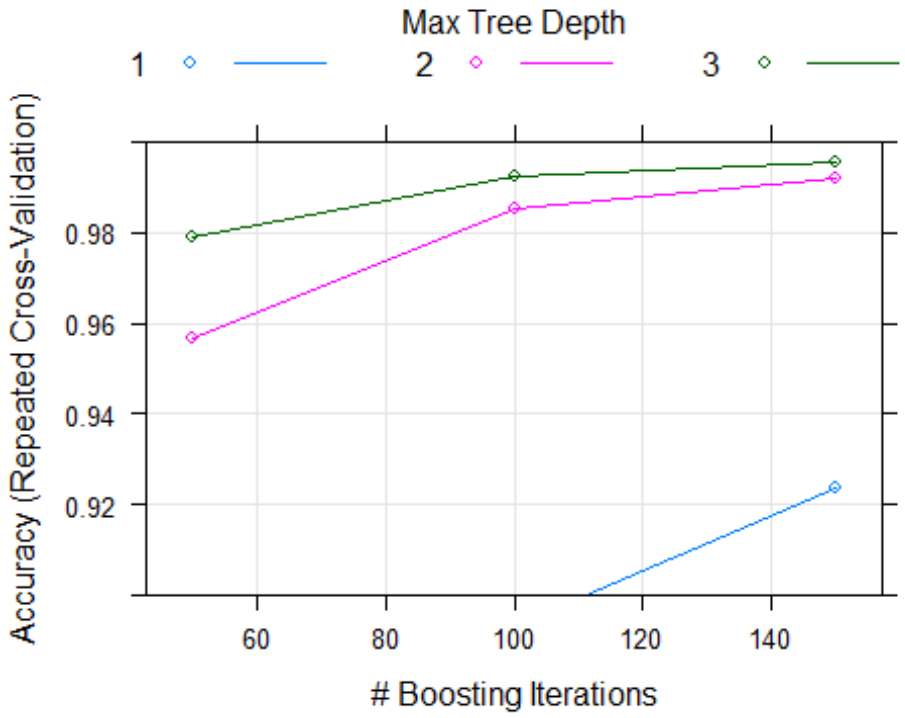
```
gbmFinMod1 <- gbmFit1$finalModel

gbmPredTest <- predict(gbmFit1, newdata=myTesting)
gbmAccuracyTest <- confusionMatrix(gbmPredTest, myTesting$classe)
gbmAccuracyTest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    4    0    0    0
##          B    0 1512    0    0    0
##          C    0    2 1365    0    0
##          D    0    0    3 1279    4
##          E    0    0    0    7 1438
##
## Overall Statistics
##
##                Accuracy : 0.9975
```

```
##                  95% CI : (0.9961, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9960   0.9978   0.9946   0.9972
## Specificity            0.9993   1.0000   0.9997   0.9989   0.9989
## Pos Pred Value         0.9982   1.0000   0.9985   0.9946   0.9952
## Neg Pred Value         1.0000   0.9991   0.9995   0.9989   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1927   0.1740   0.1630   0.1833
## Detection Prevalence   0.2850   0.1927   0.1742   0.1639   0.1842
## Balanced Accuracy      0.9996   0.9980   0.9987   0.9967   0.9981
```

```
plot(gbmFit1, ylim=c(0.9, 1))
```



Predicting Results on the Test Data

Random Forests gave an Accuracy in the myTesting dataset of 99.9%, which was more accurate that what was obtained from the Decision Trees or Generalized Boosted Regression. The expected out-of-sample error is 100-99.89 = 0.1%.

```
predictionB2 <- predict(modFitB1, testing, type = "class")
predictionB2
```

```
##  2 31  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

```
pml_write_files = function(x){
    n = length(x)
    for(i in 1:n){
        filename = paste0("problem_id_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
    }
}
```